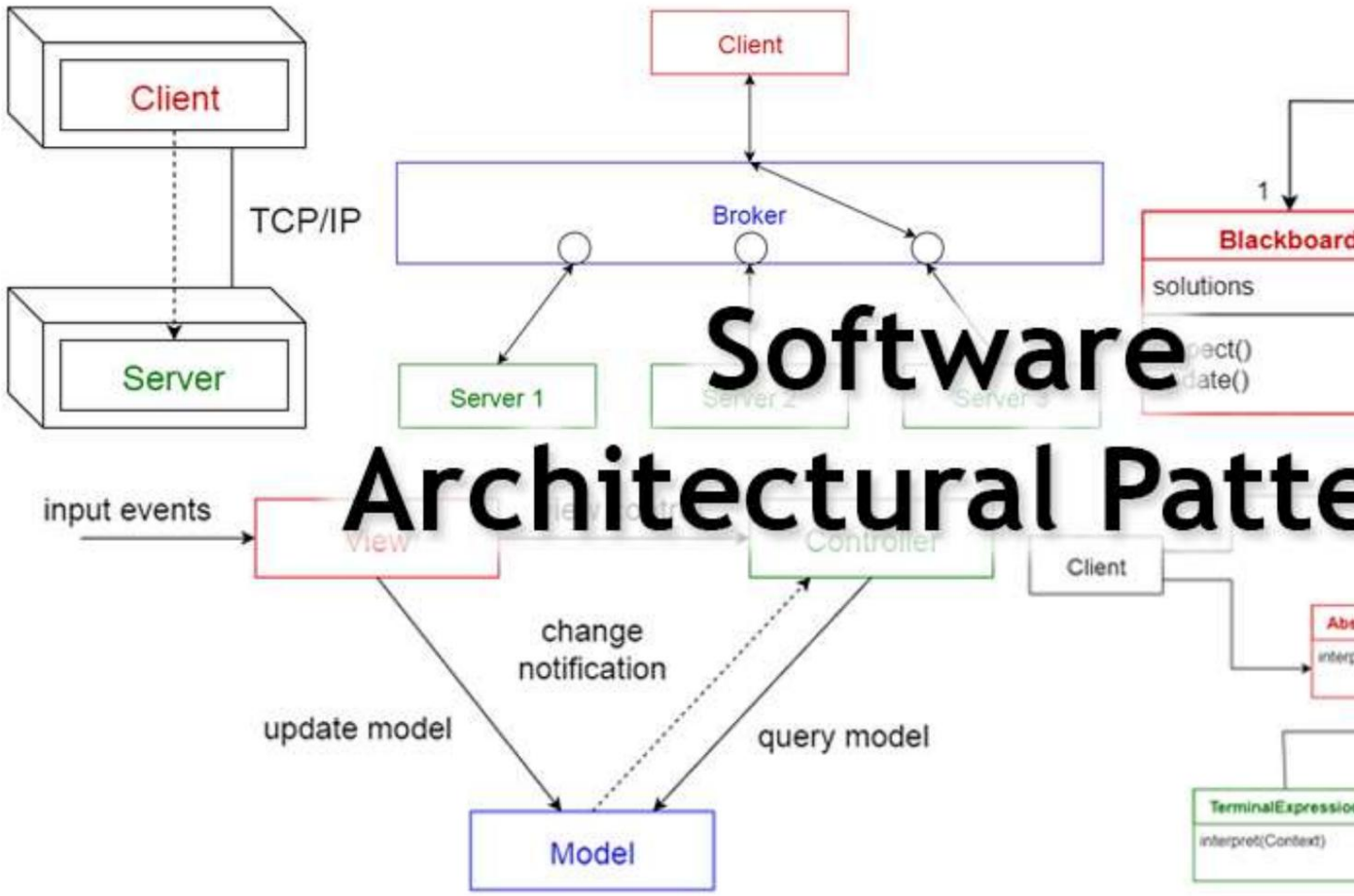
10 Phần mềm phổ biến

Các mẫu kiến trúc trong

một cái tóm lại

Bạn có bao giờ thắc mắc các hệ thống quy mô doanh nghiệp lớn đư ợc thiết kế như thế nào không? Trư ớc khi bắt đầu phát triển phần mềm lớn, chúng ta phải chọn một kiến trúc phù hợp sẽ cung cấp cho chúng ta chức năng mong muốn và các thuộc tính chất lư ợng. Do đó, chúng ta nên hiểu các kiến trúc khác nhau trư ớc khi áp dụng chúng vào thiết kế của mình.



Mẫu kiến trúc là gì?

Theo Wikipedia,
Mẫu kiến trúc là giải pháp chung, có thể tái sử dụng cho một vấn đề thường gặp trong kiến trúc phần mềm trong một bối cảnh nhất định. Mẫu kiến trúc tương tự như mẫu thiết kế phần mềm nhưng có phạm vi rộng hơn.

Trong bài viết này, tôi sẽ giải thích ngắn gọn về 10 mẫu kiến trúc phổ biến sau đây cùng cách sử dụng, ưu và nhược điểm của chúng.

1. Hoa văn nhiều lớp
2. Mô hình máy khách-máy chủ
3. Mô hình chủ-tớ
4. Mẫu ống lọc
5. Mô hình môi giới
6. Mô hình ngang hàng
7. Mô hình sự kiện-bus
8. Mô hình-view-controller
9. Mẫu bảng đen
10. Mẫu thông dịch viên

1. Hoa văn nhiều lớp

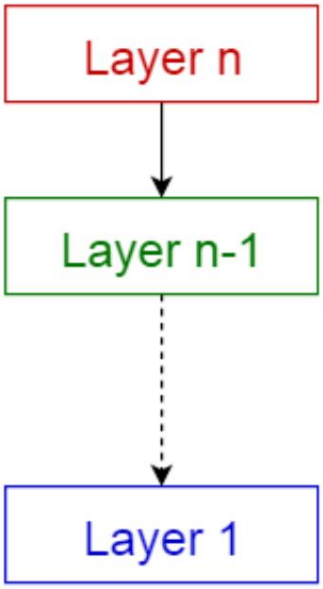
Mẫu này có thể được sử dụng để cấu trúc các chương trình có thể phân tách thành các nhóm nhiệm vụ phụ, mỗi nhiệm vụ phụ ở một mức độ trừu tượng cụ thể. Mỗi lớp cung cấp dịch vụ cho lớp cao hơn tiếp theo.

Sau đây là 4 lớp phổ biến nhất của một hệ thống thông tin chung.

- Lớp trình bày (còn gọi là lớp UI)
- Lớp ứng dụng (còn gọi là lớp dịch vụ)
- Lớp logic nghiệp vụ (còn gọi là lớp miền)
- Lớp truy cập dữ liệu (còn gọi là lớp lưu trữ)

Cách sử dụng

- Ứng dụng máy tính để bàn nói chung.
- Ứng dụng web thương mại điện tử.



Mẫu nhiều lớp

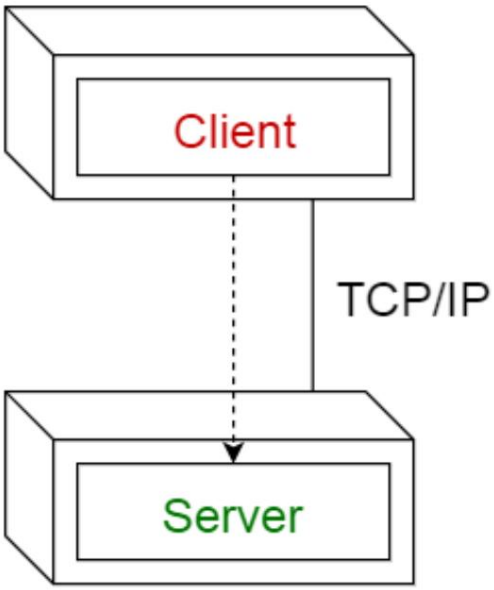
2. Mô hình máy khách-máy chủ

Mô hình này bao gồm hai bên: một máy chủ và nhiều máy khách. Thành phần máy chủ sẽ cung cấp dịch vụ cho nhiều thành phần máy khách. Máy khách yêu cầu dịch vụ từ máy chủ và máy chủ

cung cấp các dịch vụ có liên quan cho các khách hàng đó. Hơn nữa, máy chủ tiếp tục lắng nghe các yêu cầu của khách hàng.

Cách sử dụng

- Các ứng dụng trực tuyến như email, chia sẻ tài liệu và giao dịch ngân hàng.



Mô hình máy khách-máy chủ

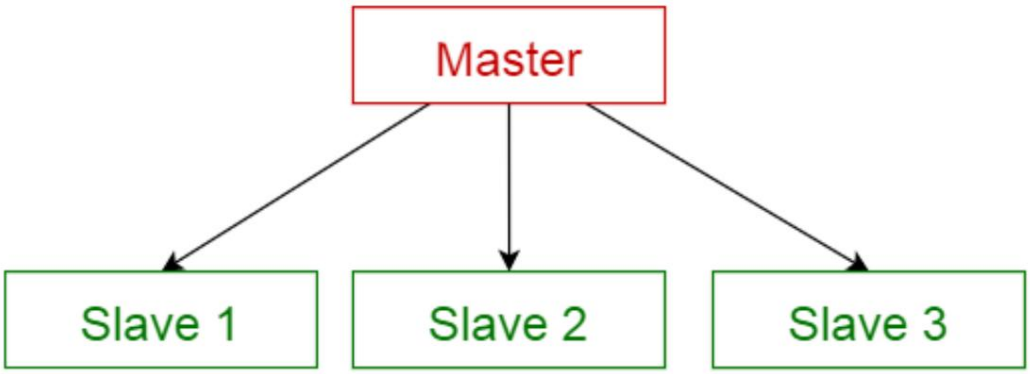
3. Mô hình chủ-tớ

Mẫu này bao gồm hai bên; master và slave. Thành phần master phân phối công việc giữa các thành phần slave giống hệt nhau và tính toán kết quả cuối cùng từ kết quả mà slave trả về.

Cách sử dụng

- Trong sao chép cơ sở dữ liệu, cơ sở dữ liệu chính được coi là nguồn có thẩm quyền và các cơ sở dữ liệu phụ được đồng bộ hóa với nguồn đó.

- Thiết bị ngoại vi được kết nối với bus trong hệ thống máy tính (ổ đĩa chính và ổ đĩa phụ).



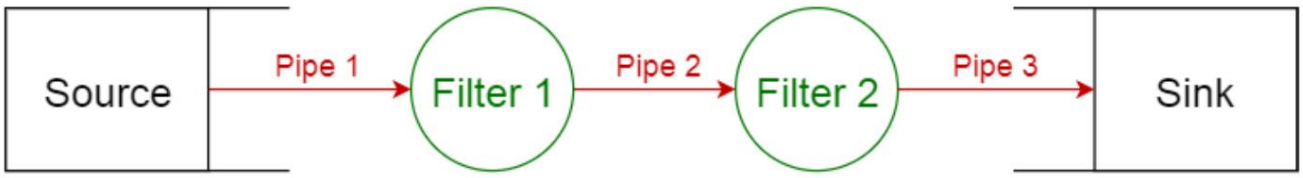
Mô hình chủ-tớ

4. Mẫu ống lọc

Mẫu này có thể được sử dụng để cấu trúc các hệ thống tạo ra và xử lý luồng dữ liệu. Mỗi bước xử lý được bao bọc trong một thành phần bộ lọc . Dữ liệu cần xử lý được truyền qua các đường ống.
Các đường ống này có thể được sử dụng để đệm hoặc đồng bộ hóa.

Cách sử dụng

- Trình biên dịch. Các bộ lọc liên tiếp thực hiện phân tích từ vựng, phân tích cú pháp, phân tích ngữ nghĩa và tạo mã.
- Quy trình làm việc trong tin sinh học.



Mẫu ống lọc

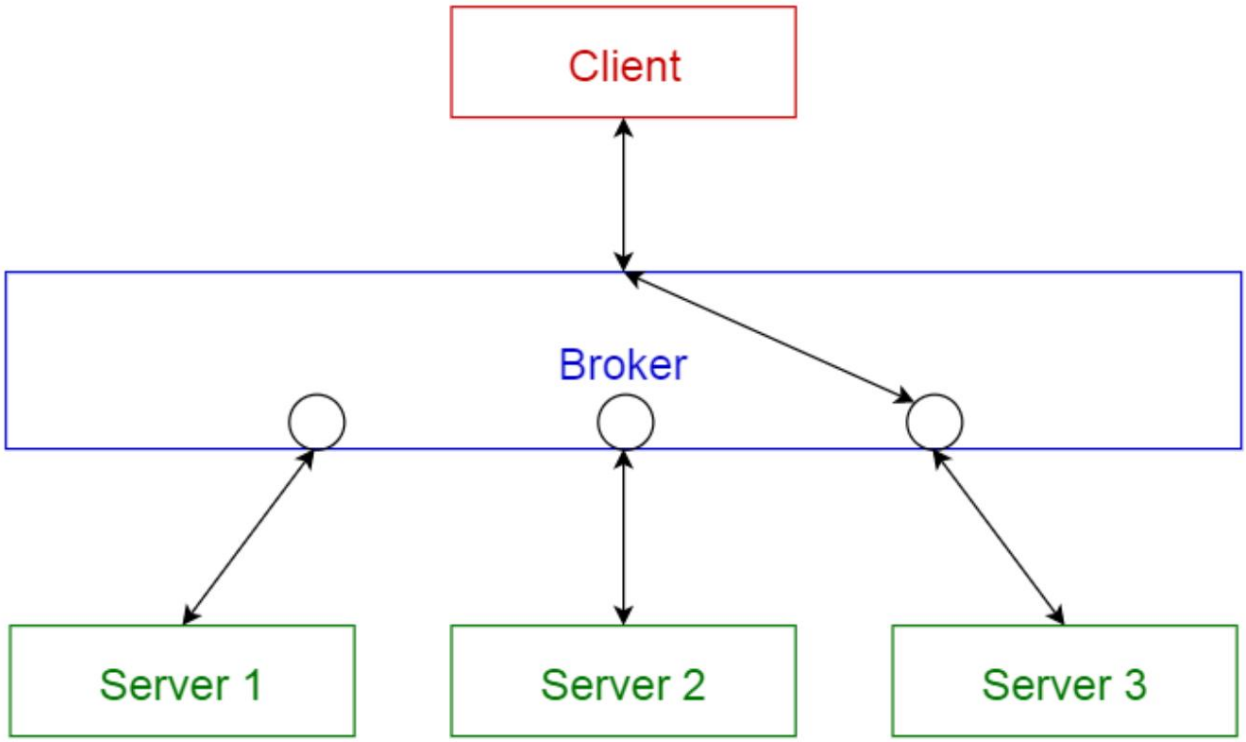
5. Mô hình môi giới

Mẫu này được sử dụng để cấu trúc các hệ thống phân tán với các thành phần tách biệt. Các thành phần này có thể tương tác với nhau bằng cách gọi dịch vụ từ xa. Một thành phần môi giới chịu trách nhiệm điều phối giao tiếp giữa các thành phần.

Máy chủ công bố khả năng của mình (dịch vụ và đặc điểm) cho một broker. Khách hàng yêu cầu một dịch vụ từ broker, sau đó broker sẽ chuyển hướng khách hàng đến một dịch vụ phù hợp từ sổ đăng ký của mình.

Cách sử dụng

- Phần mềm môi giới tin nhắn như Apache ActiveMQ, Người Apache Kafka, ThỏMQ và JBoss Messaging.



Mẫu môi giới

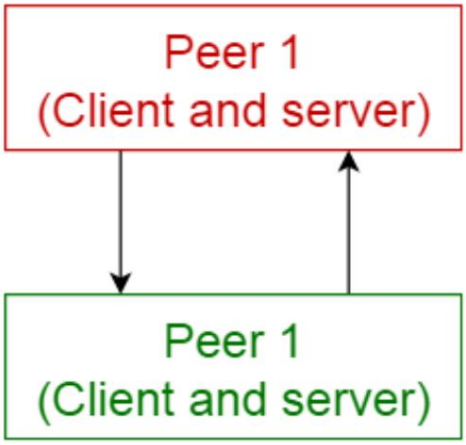
6. Mô hình ngang hàng

Trong mô hình này, các thành phần riêng lẻ được gọi là các đối tác. Các đối tác có thể hoạt động như một máy khách, yêu cầu dịch vụ từ các đối tác khác,

và như một máy chủ, cung cấp dịch vụ cho các đồng cấp khác. Một đồng cấp có thể hoạt động như một máy khách hoặc như một máy chủ hoặc cả hai, và nó có thể thay đổi vai trò của mình một cách năng động theo thời gian.

Cách sử dụng

- Mạng chia sẻ tệp như Gnutella và G2)
- Các giao thức đa phương tiện như P2PTV và PDTP.



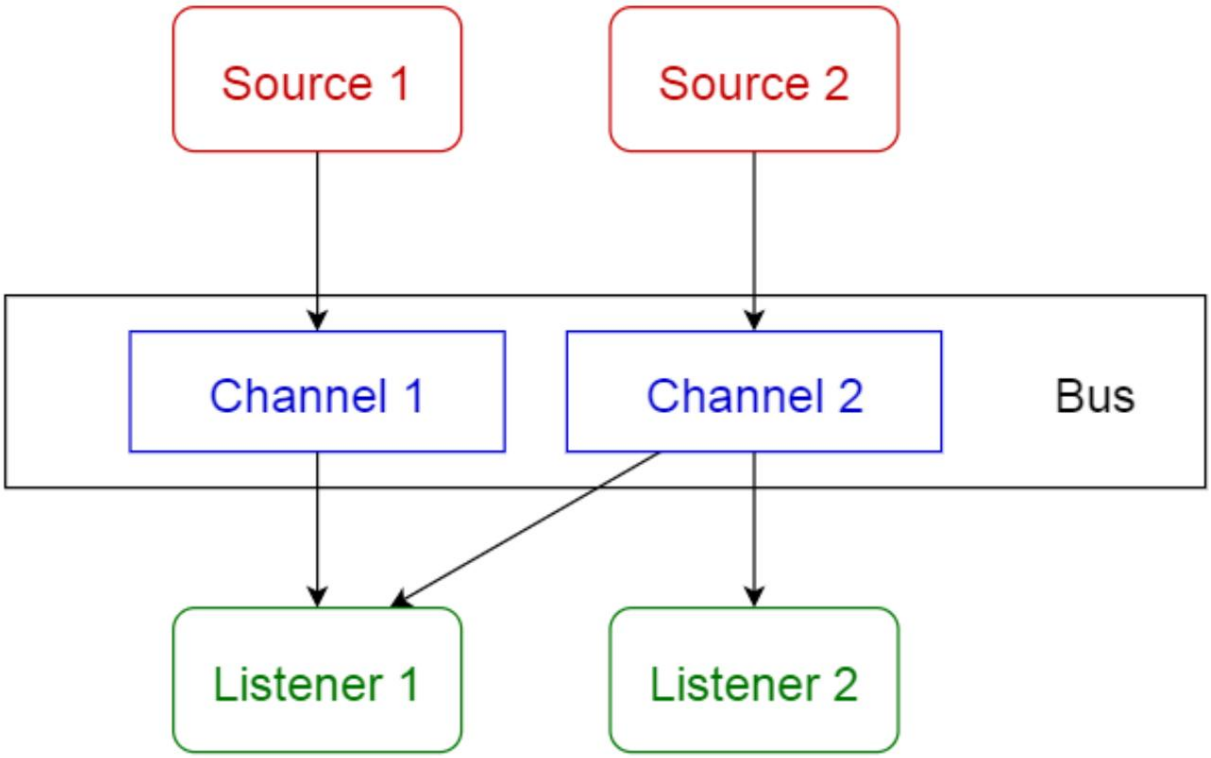
Mô hình ngang hàng

7. Mẫu event-bus Mẫu này chủ yếu xử lý các

sự kiện và có 4 thành phần chính; nguồn sự kiện, trình lắng nghe sự kiện, kênh và bus sự kiện. Nguồn phát hành tin nhắn đến các kênh cụ thể trên bus sự kiện. Trình lắng nghe đăng ký các kênh cụ thể. Trình lắng nghe được thông báo về các tin nhắn được phát hành đến kênh mà họ đã đăng ký trước đó.

Cách sử dụng

- Phát triển Android
- Dịch vụ thông báo



Mẫu sự kiện-bus

8. Mô hình-view-controller

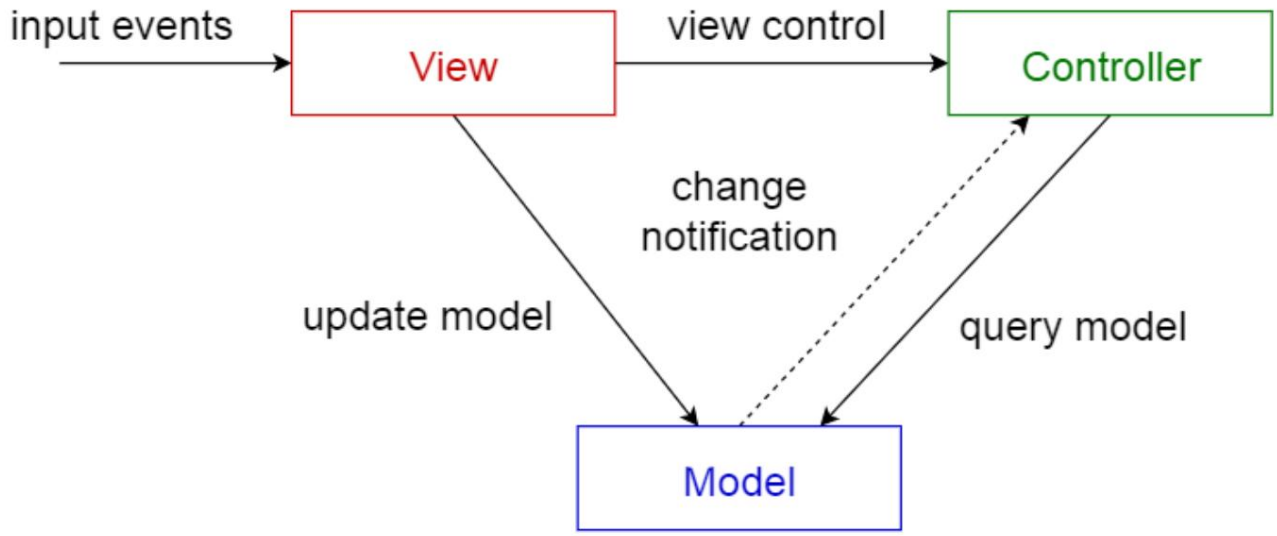
Mẫu này, còn được gọi là mẫu MVC, chia một ứng dụng tương tác thành 3 phần như sau:

1. mô hình—chứa chức năng cốt lõi và dữ liệu
2. view—hiển thị thông tin cho người dùng (nhiều hơn một chế độ xem có thể được định nghĩa)
3. bộ điều khiển—xử lý dữ liệu đầu vào từ người dùng

Điều này được thực hiện để tách biệt các biểu diễn thông tin nội bộ khỏi cách thông tin được trình bày và chấp nhận từ người dùng. Nó tách rời các thành phần và cho phép tái sử dụng mã hiệu quả.

Cách sử dụng

- Kiến trúc cho các ứng dụng World Wide Web bằng các ngôn ngữ lập trình chính.
- Các khuôn khổ web như Django và Rails.



Mô hình-xem-điều khiển

9. Mẫu bảng đen Mẫu này hữu ích cho

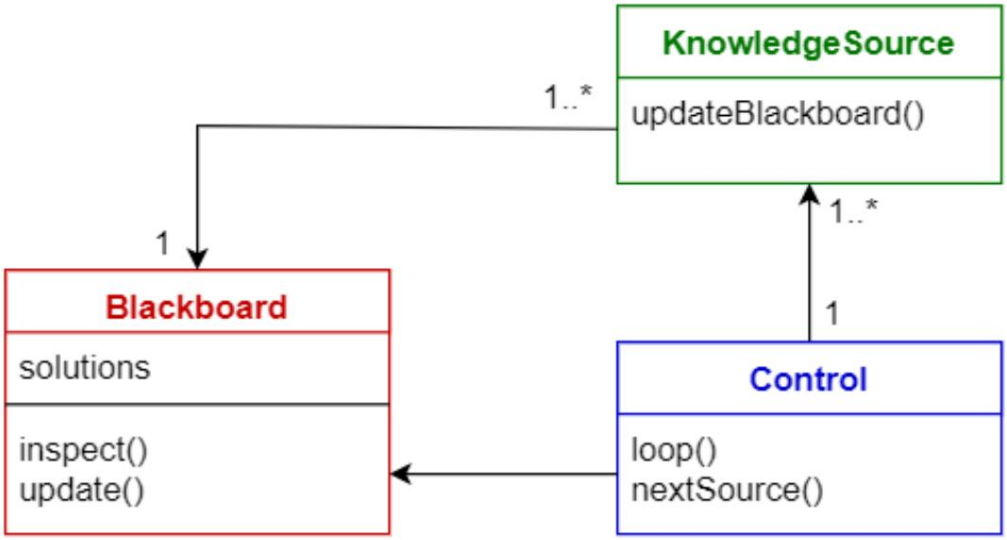
các vấn đề mà không có chiến lư ợc giải quyết xác định nào
đư ợc biết đến. Mẫu bảng đen bao gồm 3 thành phần chính.

- bảng đen– bộ nhớ toàn cục có cấu trúc chứa các đối tư ợng từ không gian giải pháp
- nguồn kiến thức –các mô-đun chuyên biệt có riêng đại diện
- thành phần điều khiển –lựa chọn, cấu hình và thực thi các mô-đun.

Tất cả các thành phần đều có quyền truy cập vào bảng đen. Các thành phần có thể tạo ra các đối tư ợng dữ liệu mới đư ợc thêm vào bảng đen.
Các thành phần tìm kiếm các loại dữ liệu cụ thể trên bảng đen và có thể tìm thấy chúng bằng cách khớp mẫu với kiến thức hiện có nguồn.

Cách sử dụng

- Nhận dạng giọng nói
- Nhận dạng và theo dõi xe
- Xác định cấu trúc protein
- Giải thích tín hiệu sonar.



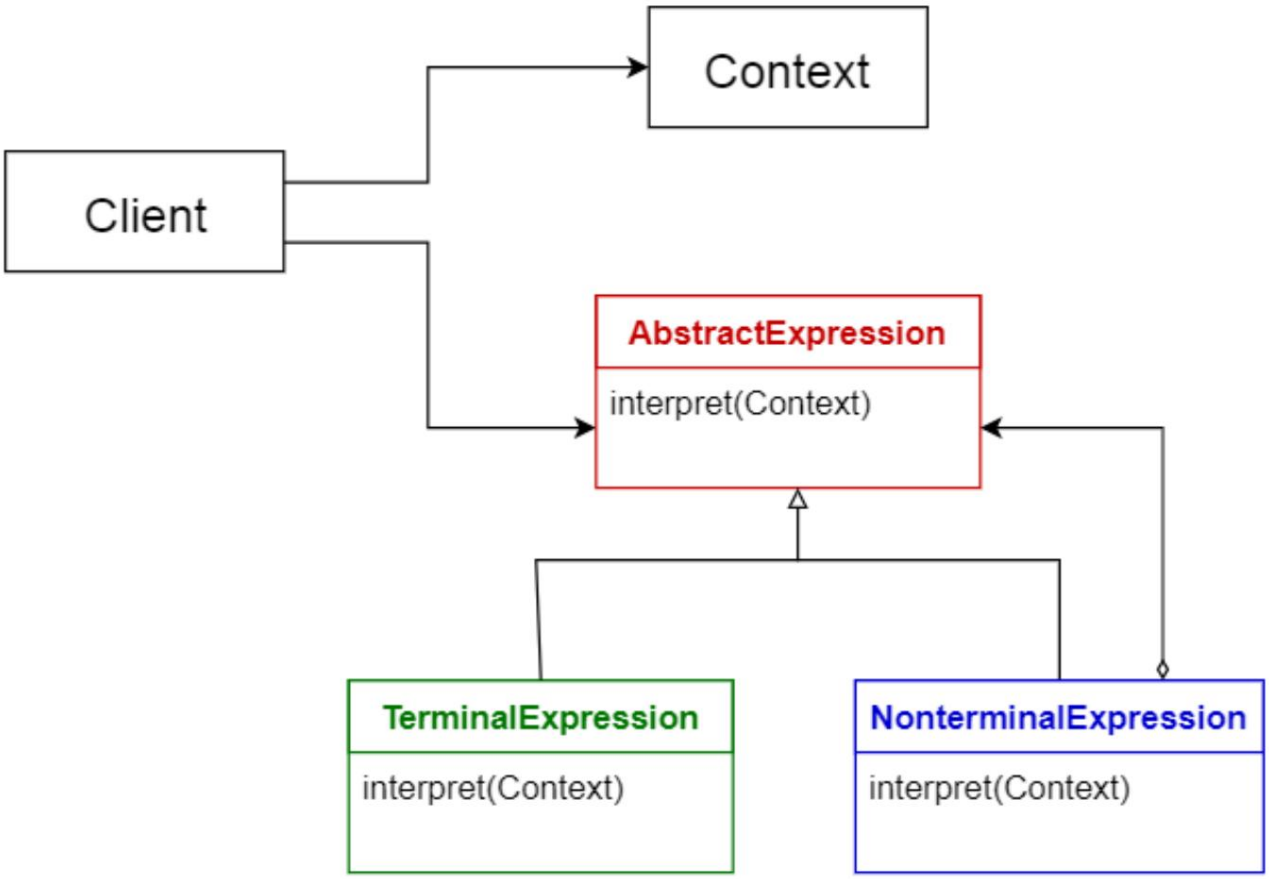
Mẫu bảng đen

10. Mẫu thông dịch viên

Mẫu này được sử dụng để thiết kế một thành phần diễn giải các chương trình được viết bằng một ngôn ngữ chuyên dụng. Nó chủ yếu chỉ định cách đánh giá các dòng chương trình, được gọi là câu hoặc biểu thức được viết bằng một ngôn ngữ cụ thể. Ý tưởng cơ bản là có một lớp cho mỗi ký hiệu của ngôn ngữ.

Cách sử dụng

- Ngôn ngữ truy vấn cơ sở dữ liệu như SQL.
- Ngôn ngữ được sử dụng để mô tả giao thức truyền thông.



Mẫu thông dịch viên

So sánh các mẫu kiến trúc

Bảng dư ới đây tóm tắt ư u và như ợc điểm của từng mẫu kiến trúc.

Name	Advantages	Disadvantages
Layered	A lower layer can be used by different higher layers. Layers make standardization easier as we can clearly define levels. Changes can be made within the layer without affecting other layers.	Not universally applicab Certain layers may have
Client-server	Good to model a set of services where clients can request them.	Requests are typically ha Inter-process communic representations.
Master-slave	Accuracy - The execution of a service is delegated to different slaves, with different implementations.	The slaves are isolated: The latency in the maste real-time systems. This pattern can only be
Pipe-filter	Exhibits concurrent processing. When input and output consist of streams, and filters start computing when they receive data. Easy to add filters. The system can be extended easily. Filters are reusable. Can build different pipelines by recombining a given set of filters	Efficiency is limited by th Data-transformation ove
Broker	Allows dynamic change, addition, deletion and relocation of objects, and it makes distribution transparent to the developer.	Requires standardizatior
Peer-to-peer	Supports decentralized computing. Highly robust in the failure of any given node. Highly scalable in terms of resources and computing power.	There is no guarantee ab Security is difficult to be Performance depends o
Event-bus	New publishers, subscribers and connections can be added easily. Effective for highly distributed applications.	Scalability may be a prob
Model-view-controller	Makes it easy to have multiple views of the same model, which can be connected and disconnected at run-time.	Increases complexity. M
Blackboard	Easy to add new applications. Extending the structure of the data space is easy.	Modifying the structure May need synchronizati
Interpreter	Highly dynamic behavior is possible. Good for end user programmability. Enhances flexibility, because replacing an interpreted program is easy.	Because an interpreted performance may be an